**from \_\_future\_\_ import** print\_function

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**5.1 Create DataFrame**

columns = ['name', 'age', 'gender', 'job']

user1 = pd.DataFrame([['alice', 19, "F", "student"],

['john', 26, "M", "student"]],

columns=columns)

user2 = pd.DataFrame([['eric', 22, "M", "student"],

['paul', 58, "F", "manager"]],

columns=columns)

user3 = pd.DataFrame(dict(name=['peter', 'julie'],

age=[33, 44], gender=['M', 'F'],

job=['engineer', 'scientist']))

**print**(user3)

import matplotlib.pyplot as plt

# Load the image

data = plt.imread('bricks.png')

# Display the image

plt.imshow(data)

plt.show()

# Set the red channel in this part of the image to 1

data[:10,:10,0] = 1

# Set the green channel in this part of the image to 0

data[:10,:10,1] = 0

# Set the blue channel in this part of the image to 0

data[:10,:10, 2] = 0

# Visualize the result

plt.imshow(data)

plt.show()

OpenCV Python Tutorial | Creating Face Recognition System And Motion Detector Using OpenCV | Edureka

import cv2

# Load the image

face\_cascade = cv2.cascadeClassifier("haarcascade\_frontalface\_default.xml")

img = cv2.imread ("D:\\phyton\\New folder\\mario.jpg")

gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray\_img, scaleFactor = 1.05,

minNeigbors=5)

print(type(faces))

print(faces)

for x,y,w,h in faces:

img = cv2.rectangule(img, (x,y) , (x+y,w+h) , (0,255,0) , 3)

cv2.imshow("legend",img)

cv2.waitKey(0)

cv2.destroyAllWindows()

import cv2

# Load the image

face\_cascade = cv2.CascadeClassifier("D:\\phyton\\New folder\\haarcascade\_frontalface\_default.xml")

img = cv2.imread ("D:\\phyton\\New folder\\mario.jpg")

gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray\_img, scaleFactor = 1.05,

minNeighbors=5)

print(type(faces))

print(faces)

for x,y,w,h in faces:

img = cv2.rectangule(img, (x,y) , (x+y,w+h) , (0,255,0) , 3)

cv2.imshow("legend",img)

cv2.waitKey(0)

cv2.destroyAllWindows()

# How to do Object Detection with OpenCV [LIVE

D:\\phyton\\New folder\\haarcascade\_frontalface\_default.xml

**QR code generator**

# Import QR Code library

import qrcode

# Create qr code instance

qr = qrcode.QRCode(

version = 1,

error\_correction = qrcode.constants.ERROR\_CORRECT\_H,

box\_size = 15,

border = 4,

)

# The data that you want to store

data = "zunairwahid1@gmail.com"

# Add data

qr.add\_data(data)

qr.make(fit=True)

# Create an image from the QR Code instance

img = qr.make\_image()

# Save it somewhere, change the extension as needed:

# img.save("image.png")

# img.save("image.bmp")

# img.save("image.jpeg")

img.save("picture.jpg")

NLP

# Import necessary modules

from nltk.tokenize import sent\_tokenize

from nltk.tokenize import word\_tokenize

# Split scene\_one into sentences: sentences

sentences = sent\_tokenize(scene\_one)

# Use word\_tokenize to tokenize the fourth sentence: tokenized\_sent

tokenized\_sent = word\_tokenize(sentences[3])

# Make a set of unique tokens in the entire scene: unique\_tokens

unique\_tokens = set(word\_tokenize(scene\_one))

# Print the unique tokens result

print(unique\_tokens)

# Split the script into lines: lines

lines = holy\_grail.split('\n')

# Replace all script lines for speaker

pattern = "[A-Z]{2,}(\s)?(#\d)?([A-Z]{2,})?:"

lines = [re.sub(pattern, '', l) for l in lines]

# Tokenize each line: tokenized\_lines

tokenized\_lines = [regexp\_tokenize(s, "\w+") for s in lines]

# Make a frequency list of lengths: line\_num\_words

line\_num\_words = [len(t\_line) for t\_line in tokenized\_lines]

# Plot a histogram of the line lengths

plt.hist(line\_num\_words)

# Show the plot

plt.show()

**TEXT CLASSIFICATION AND CLARIFICATION**

**import numpy as np**

**import re**

**import nltk**

**from sklearn.datasets import load\_files**

**nltk.download('stopwords')**

**import pickle**

**from nltk.corpus import stopwords**

**movie\_data = load\_files(r"D:\\python")**

**X, y = movie\_data.data, movie\_data.target**

**documents = []**

**for sen in range(0, len(X)):**

**# Remove all the special characters**

**document = re.sub(r'\W', ' ', str(X[sen]))**

**# remove all single characters**

**document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)**

**# Remove single characters from the start**

**document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)**

**# Substituting multiple spaces with single space**

**document = re.sub(r'\s+', ' ', document, flags=re.I)**

**# Removing prefixed 'b'**

**document = re.sub(r'^b\s+', '', document)**

**# Converting to Lowercase**

**document = document.lower()**

**# Lemmatization**

**document = document.split()**

**document = [stemmer.lemmatize(word) for word in document]**

**document = ' '.join(document)**

**documents.append(document)**

**from sklearn.feature\_extraction.text import CountVectorizer**

**vectorizer = CountVectorizer(max\_features=1500, min\_df=5, max\_df=0.7, stop\_words=stopwords.words('english'))**

**X = vectorizer.fit\_transform(documents).toarray()**

**Tf= (Number\_of\_Occurrence\_of\_a\_word)/(Total\_words in the\_document)**

**IDF(word) = Log((Total number of documents)/(Number of documents containing the word))**

**from sklearn.feature\_extraction.text import TfidfTransformer**

**tfidfconverter = TfidfTransformer()**

**X = tfidfconverter.fit\_transform(X).toarray()**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)**

**classifier = RandomForestClassifier(n\_estimators=1000, random\_state=0)**

**classifier.fit(X\_train, y\_train)**

**y\_pred = classifier.predict(X\_test)**

**from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score**

**print(confusion\_matrix(y\_test,y\_pred))**

**print(classification\_report(y\_test,y\_pred))**

**print(accuracy\_score(y\_test, y\_pred))**

**with open('text\_classifier', 'wb') as picklefile:**

**pickle.dump(classifier,picklefile)**

**with open('text\_classifier', 'rb') as training\_model:**

**model = pickle.load(training\_model)**

**y\_pred2 = model.predict(X\_test)**

**print(confusion\_matrix(y\_test, y\_pred2))**

**print(classification\_report(y\_test, y\_pred2))**

**print(accuracy\_score(y\_test, y\_pred2))**

import nltk

import random

from nltk.corpus import movie\_reviews

nltk.download()

moviedir = (r"D:\\python")

documents = [(list(movie\_reviews.words(fileid)), category)

for category in movie\_reviews.categories()

for fileid in movie\_reviews.fileids(category)]

random.shuffle(documents)

print(documents[1])

all\_words = []

for w in movie\_reviews.words():

all\_words.append(w.lower())

all\_words = nltk.FreqDist(all\_words)

print(all\_words.most\_common(15))

print(all\_words["stupid"])

documents = [(list(movie\_reviews.words(fileid)), category)

for category in movie\_reviews.categories()

for fileid in movie\_reviews.fileids(category)]

print(all\_words.most\_common(15))

print(all\_words["stupid"])